AIRBUD

Artificially intelligent control test bed that detects and tracks objects.

EP 497, Spring 2024

Jose Castelblanco, Conor Metz, Maddox Morrison

ACKNOWLEDGEMNTS

We thank the Office of Undergraduate Research for giving us a SPARK Grant to assist in our project. We would also like to thank Dr. Sergey Drakunov and the entire Engineering Physics Propulsion Laboratory for their mentorship and resources.

Executive Summary

This report presents the development of an artificially intelligent quadcopter, named AIRBUD, designed as a control test bed for the bigger goal of tapping into the potential with quadcopters and object detection. The project's goal is to create a quadcopter that can behave as a test bed for this larger goal. AIRBUD, equipped with AI, serves as a dynamic platform for testing and validating control algorithms for autonomous object detection and tracking. The report covers various aspects of the project, including the societal and market need for space debris mitigation, the specific engineering challenges and solutions provided by AIRBUD, the system's design and implementation, and the results achieved.

The report outlines the technical details of AIRBUD's development, including the integration of artificial intelligence, robotics, and control systems for object detection, maneuverability, tracking, and the challenges faced in refining these systems. System constraints ensure optimal performance, with Simulation in Hardware (SIH) validating changes. Data obtained from a variety of flight tests is presented showing effective completion of the initial expectations and requirements. AIRBUD successfully completed desired objectives and provided good data that was able to fully analyze.

Table of Contents

Table of Figures

Project Description

I. Introduction

Quadcopters are becoming more relevant and integral parts of our current society. Every single industry, from defense, to security, to even agriculture, can take advantage of quadcopters and use them to implement innovative solutions. When combined with object tracking and identification, these systems can evolve further, incorporating artificial intelligence into everyday life. This design review serves as an exploration of the AIRBUD (Artificially Intelligent Route Building Unsupervised Drone) project, the senior design course's focus. AIRBUD serves as a smaller component that can be universally applied to any industry. It focuses on creating a artificially intelligent operating quadcopter that combines elements from controls and robotics together. Working together these elements create a control testbed that allows for object identification and object tracking. As this becomes more widespread, research into object identification and tracking integrated into quadcopters will be more readily available. Society is still at the beginning stages of these advancements and current research is focused on neural network integration and creating a fully operating artificially intelligent system. This project aims to increase understanding and practicality in this field. To design this project a Jetson Orin NX, ZedX mini camera, and CubeOrangePlus flight controller were integrated together to achieve desired objectives. AIRBUD serves as a control testbed on an artificially intelligent operating quadcopter that can detect and track desired objects.

II. System Need

This project addresses the growing industry, as stated, of using artificial intelligence incorporated with quadcopters. One very important area where this can be seen is with the security industry. Businesses, homes, and public areas have started to use more advanced protection systems to keep themselves and their customers safe. AIRBUD fulfills this need and can be implemented into this domain as it is focused on detecting suspicious persons or activities. By having automatic detection of these activities from just a generic camera, homeowners or head security officers can be alerted to dangerous situations preventing tragedy. A market need in this area is very clearly seen as companies such as RING have already started to implement widespread detection. AIRBUD’s systems would allow more advanced detections to take place, or it can be applied to an aerial vehicle. Many of these already implemented systems use basic neural networks for their detection and AIRBUD’s convolutional neural networks would provide faster and more accurate results.

Another important area where this can be implemented is with space debris. The societal need to address space debris stems from its increasing threat to space activities and satellite operations, driven by the proliferation of satellites from various entities. This congestion raises collision risks and endangers critical systems like GPS. The escalating debris population heightens the risk of the "Kessler Syndrome," impacting space exploration and vital Earth services. This necessitates strategies for active debris removal to ensure sustainable space use. The market demand is fueled by growing investments in satellite tech, with governments seeking mitigation solutions to protect satellites and maintain services. Stakeholders, including satellite operators, space agencies, and private firms, are crucial in developing efficient debris removal tech and establishing global cooperation and policies for responsible space use. The AIRBUD system we have designed addresses specific needs within the realm of space debris cleanup by serving as a test bed for advanced control algorithms. One of the key challenges in space debris mitigation is the development and validation of robust control systems capable of efficiently identifying, tracking, and interacting with space debris. AIRBUD, functioning as a quadcopter with artificial intelligence (AI), fulfills the need for a dynamic platform that can simulate and assess control algorithms in a controlled environment.

Current solutions for space debris removal face several functional and operational deficiencies that require improvement in capabilities and performance. One significant challenge is the lack of adaptability and versatility in tracking and capturing different types of debris. Existing solutions struggle to handle the wide range of debris shapes, sizes, and materials, leading to limitations in their applicability. Improving the capability to address this diversity is crucial to ensuring a comprehensive and effective space debris cleanup. Operational deficiencies also arise from limitations in the maneuverability and agility of current systems. Space debris often follows unpredictable trajectories, and the ability to dynamically adjust and optimize the trajectory of debris removal systems is essential. Existing solutions have yet to implement effective measures to address this. Additionally, autonomy in decision-making is a critical aspect that requires enhancement. Many existing systems heavily rely on ground-based control, which introduces communication delays and limits the ability to respond promptly to dynamic debris scenarios. Implementing advanced artificial intelligence and autonomy in decision-making processes will empower debris removal systems to operate more independently, making split-second decisions based on real-time data.

A last area of industry application can be seen with defense. Currently in the Ukraine-Russia conflict, the use of drones and specifically quadcopters have exploded. Prior conflicts have mostly been dependent on traditional troop movement along with land and aerial vehicles. Due to the increase in drone technology and AI advancements, these small devices have now taken the forefront. Additionally, these drones are much cheaper compared to traditional devices and do not cost human lives. Operations will one day be fully carried out by devices like drones that can use object identification and tracking. The current issues stem from having effective object detection algorithms that can track troop movement and enemy supply chains. Required resources need extremely accurate data or it could cost the lives of hundreds of friendly troops. AIRBUD aims to create a top-of-the-line object detection system that will be able to save lives in the future. These systems are very new and are not well documented for current use. Videos from conflict however show that they are operational, and the operational capacity of these quadcopters can be greatly improved in both maneuverability and detection accuracy.

To align with AIRBUD's mission of serving as a test bed for control algorithms in many industries, there is a set of goals and measures of effectiveness established. These are requirements that AIRBUD must complete to call the system a success. All these goals are actionable and provide measurable metrics that help AIRBUD operate as an artificially intelligent operating quadcopter that can detect and track desired objects.

Maneuverability pertains to the first requirement and measurement of effectiveness for AIRBUD. The quadcopter must be able to quickly maneuver to find a desired object while giving an accurate horizontal and vertical position in space. The following requirements have been laid out for this system.

- More than 3500 grams of thrust
- Accurate horizontal position by 5 cm
- Accurate vertical position by 5 cm
- Yaw angular rate over 2.0 rad/s

AIRBUD must also be able to accurately detect desired objects, a ball and person, to say that the system was a success. The algorithm YOLOV8 is implemented into AIRBUD and serves as the main object detection source. We require the following parameters.

- Precision 90%
- Recall 65%
- F1 Score 65%
- Confidence level 60%

The recall and F1 score are at 65% due to the large number of false positives that we expect the algorithm to detect. YOLOV8 is designed for larger objects and the small ball that is our main object can be easily misconstrued as other objects in the image frame. The confidence level is at 60% as it the optimal range for detections of the ball.

Object tracking allows the drone to alter its yaw angle to follow the ball as it moves across the quadcopter. An algorithm must be designed to Implement an object tracking algorithm that will respond to the ball and follow it by changing the yaw angle. For this to be a success the following parameters must be met.

- Motor response time under 1 second
- Settling time of 0.5 seconds
- Max overshoot no more than 20%
- Reject disturbances caused by wind speeds up to 10 mph

Lastly, AIRBUD must have an offboard mode which allows both manual and automatic flight of the quadcopter. All operations will be done safely with appropriately taken measures. Prioritizing safety by putting in an offboard switch ensures that the user always has control.

- Emergency offboard switch setup
- Low battery voltage of 12.8 V
- Implemented fuses to protect the camera and jetson
- Standardized test setup that can be reciprocated

III. Problem Definition

This specific project involves the development of an artificially intelligent quadcopter designed as a control test bed for the identification, tracking, and prediction of an object's path. The project integrates machine learning with control theory methodologies. To attain this ultimate objective, specific expectations must be clearly defined and met. This section will delve into the details of maneuverability, object detection, path prediction, object tracking, and safety considerations.

AIRBUD is designed to swiftly respond, and track detected objects, requiring it to function as a manually operable RC drone. The CubeOrange plus must transmit signals to the motors within a 20 ms timeframe to facilitate rapid drone movement. From image detection however, it must take less than one second for the reaction to occur. Ensuring more than 3500 grams (about 7.72 lb) of thrust from the motors is essential to move all components effectively. For optimal directional changes, the roll, pitch, and yaw must provide a sufficient angle of motion. The drone should have the capability to rotate a full 360 degrees, enabling a comprehensive range for tracking. During operations, the roll and pitch angles are limited to a maximum of 20 degrees of tilt to facilitate rapid movement without compromising the center of mass balance. Striking a balance is crucial, as too small a tilt angle might result in delayed responses to changing object paths, while excessive tilt could render the drone unstable in flight. Likewise, Accurate position must be established, with vertical and horizontal errors of up to 5 cm from desired setpoint, to ensure proper maneuverability.

Utilizing YOLOv8s as our object detection algorithm, coupled with distance computed using the disparity between images captured by the left and right cameras of the Zedx stereo setup, AIBUD must be able to detect and track objects. To aid with the detection, IMU (inertial measurement unit) data and neural networks are used to give information regarding current positions. IMUs are typically made up of three different sensors. Neural networks are computational models that act as machine learning programs. They act in a way similar to that of the human brain, hence giving them the name ‘neural.’ Biological processes of thought, deduction, and decision-making are being replicated by these neural networks which make them so impactful and important for future generations. They consist of interconnected layers of nodes or neurons that learn and make information-based decisions. These artificial brains are then used in deep learning environments where a built machine can take information, come up with conclusions regarding the information, and decide based on the supplied data. To aid object detection on AIRBUD, convolutional neural networks are used which gives us desired detection characteristics. Specifically, these networks are used to assist with matrix multiplication, linear algebra, and pattern recognition by mapping out image areas. AI image recognition uses these neural networks to assist with object detection, classification, and even facial recognition. They have several convoluted layers in-between input and output which aids with the image detection process.

The first IMU sensor is a gyroscope which measures the angular velocity of the drone. It determines how fast and in what direction the quadcopter is rotating while presenting that data to the Ground control station. It operates using advanced Micro-electromechanical systems

(MEMS) outside this project's scope. Accelerometers measure the rate at which things speed up and slow down. These also use MEMS, but they are more simplified for these measurements. Lastly, IMUs always have magnetometers to act like a digital compass for the system Magnetometers use the Hall Effect to determine exactly what the B field is in the surrounding area of the drone. These three sensors, combined with other applied sensors on the quadcopter help determine accurate positions and set the boundaries for the object tracking. All sensor measurements are brought into an extended Kalman filter (EKF) which creates a prediction of the current state of the drone. The sensor measurements are each given individualized weights, determining how effective or accurate the sensor is. This is done through statistical methods in the EKF, and it picks the most desirable sensor measurements to use. This process is iterative and is continuously updated as new measurements come into the system.

Through these sensors, the custom flight control algorithm will converge with the desired yaw rate with a settling time of .5 seconds. The max overshoot of the response should not be more than 20%. The custom controller should be capable of rejecting disturbances caused by wind speeds of 10 mph.

Ensuring safety is the utmost priority for this project due to the potential severe consequences associated with mishandling a quadcopter. To address this, the drone will feature an emergency switch (Switch 6 on the controller) enabling an instantaneous transition between automatic and manual control. This is the offboard mode previously mentioned and allows quick transition between independent versus manual control. In the event of any automatic control issues, a designated manual operator will promptly assume expert control and initiate a controlled landing to halt the flight. Additionally, to mitigate the risk of false positive object detection by the AI system, emergency switch 6 will again allow a manual operator to take control. To further minimize safety risks, the drone will only be operated under specific conditions, including a voltage above 12.8 V and ideal weather conditions. Unfavorable weather elements such as excessive wind, rain, or disturbances will be grounds for prohibiting any drone flights. Fuses will be implemented to ensure that the most important sensors remain undamaged and are protected. Simplifying the process further, all testing will be done following a strict procedure to make sure that no mistakes are made.

These problem statements give us four milestones or pillars that are instrumental to the project's foundation/success.

1. Complete the communication process between the Jetson and the CubeOrangePlus and the camera through ROS so that messages can be received by the flight controller.
2. Train the neural network object detection system through the mini Zedx camera to fully recognize the ball, and humans.
3. Use information provided to the Jetson Orin NX to complete a control objective allowed by the RC Controller.
4. To identify and track a ball with accurate yaw rate using all concurrent systems while having a trajectory predicting algorithm.

IV. Solution

This research initiative builds on the accomplishments of a previous project conducted in the summer of 2023. During that project, we successfully engineered an omnidirectional robot with the capability to identify and catch a thrown ball. The project involved the implementation of transfer learning, where a pre-trained object detection model was fine-tuned to meet specific requirements. This process included the development of a dataset comprising thousands of images, which were annotated and utilized for the supervised training of a Convolutional Neural Network (CNN). The trained CNN demonstrated precise object position predictions in new data, such as live feeds, showcasing real-time inferencing capabilities. This success hinted at potential applications in fields like industrial diagnostics and robotics. Despite facing hardware constraints, the project validated the viability of AI-driven object capture and laid the foundation for future AI-centric engineering solutions. Building off of these achievements, our current proposal aims to push the boundaries of our knowledge and implementation. Shifting our focus from a ground-based vehicle, we are now dedicated to designing a more advanced drone-based system, leveraging newer and more capable hardware and software. The Spark Grant, requesting $1000 for the acquisition of the Zedx mini camera, was successfully secured to facilitate this project. The funding allows us to purchase the camera, and as a result, we presented our progress at Discovery Day, acknowledging the support received from the Department of Undergraduate Research.

<u>Components:</u> This section composes of all the systems

<u>Chassis:</u> The chassis, weighing between 500 and 510 grams and measuring 354x354x268 mm, is constructed from carbon fiber, known for its lightweight and high-strength properties. The chassis provides the necessary structural integrity for the quadcopter.

<u>Equipment Mounts:</u> Various components are securely attached to the chassis using a combination of methods. For instance, the Jetson Orin Nano and camera are attached via 3D printed casings, motors are fastened with M3 screws, and batteries are secured using Velcro straps.

<u>Motors:</u> The drone's propulsion system relies on four brushless electronic motors that work in tandem to achieve stable flight. These antigravity motors operate at 470 KV and each individual motor provides 4.2 kg of thrust.

<u>Propellers:</u> QWinOut propellers, made of lightweight and sturdy carbon fiber, are employed to provide lift when rotated by the motors. With a size of 15.5” x 5.5”, each propeller weighs 20 grams, contributing to the overall efficiency of the actuation system.

<u>CubeOrange Plus Flight Controller:</u> The CubeOrange Plus flight controller serves as the control center, receiving commands from the RC, Jetson onboard computer, and ground station. Equipped with various sensors, including an accelerometer, barometer, magnetometer, and gyroscope, the flight controller ensures stable flight.

GPS: Here 3 GPS unit is essential for autonomous navigation, providing precise location data to the flight controller. Weighing 54g, the GPS unit ensures accurate positioning with a 1 cm radius.

TF Luna Lidar: The Lidar is located on the bottom of the drone and provides a vertical estimate for the current position. It has an acceptance angle of 2.3 degrees and does not vary much from a straight path. It uses a vertical cavity surface emitting laser to calculate its position.

Dampeners: The dampeners are placed under the CubeOrangePlus and the Jetson to help protect them from natural in-flight vibrations.

Jetson Orin NX: The Jetson Orin NX serves as the onboard computer running AI algorithms. Composed of a 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores and an 8-core NVIDIA Arm Cortex A78AEv8.2 64-bit CPU. The GPU has 16GB of RAM, which is shared with the CPU, capable of 100 Tera operations per second.

Camera: the Zed X mini stereo camera captures images at high-resolution HD1200 at 60 FPS, with a pixel size at this resolution of 0.003 mm. Likewise, the focal length at this resolution is 2.2 mm or 737 pixels The camera and its intrinsic properties can be accessed by the ZED SDK and ZedWrapper for ROS utilization.

Stereo Capture: The Zed X camera, with dual lenses, captures high-definition 3D video with a wide field of view. Images are processed by the ISP of the Jetson and provided in RGB format on the host.

Depth Sensing: Depth maps captured by the Zed X store distance values for each pixel, creating a 3D point cloud. This point cloud represents the external surface of the scene and can contain color information.

Positional Tracking: The Zed X uses visual tracking to understand its movement, updating its position and orientation up to the frame rate of the camera. This information is referred to as the camera 6DoF pose.

Power System: Figure 1 shows the power system for the Jetson and camera. Designed for protection from short circuits, overload, and voltage spikes, that could be caused by 4s LiPO battery, by utilizing fast-blow fuses and DC to DC buck converters described below. Figure 2 shows the power system for the drone flight controllers and motors. This power system consists of a 6s LiPO battery followed by a voltage step down/power distribution circuit that provides 5V to the flight controller and 22.2V to the ESCs.

4s 6AHr Battery: 4 cells in series Lithium Polymer (LiPO) type battery with a nominal voltage of 14.8 volts, and a capacity of 6000 Milliamp Hours. With its high discharge rate of 50C, it can theoretically supply 300 amps in current. Total energy storage of 88.8 WH.

6s 1.5 AHr Battery: 6 cells in series Lithium Polymer (LiPO) type battery with a nominal voltage of 22.2 volts, and a capacity of 1500 Milliamp Hours. With its high discharge rate of 100C, it can theoretically supply 150 amps in current. Total energy storage of 33.3 WH.

40A ESCs: T-motor Air 40A 600Hz 2-6s ESC no BEC.

Buck Converters: 180 KHz fixed frequency PWM buck (step-down) DC/DC module, capable of driving a 5A load with high efficiency, low ripple and line and load regulation. Input voltage can be ranged from 4-38V while output can be set between 1.25 to 36V.

Fuses: Fast-Blow glass fuses for short circuit and overload protection.

Diagram:

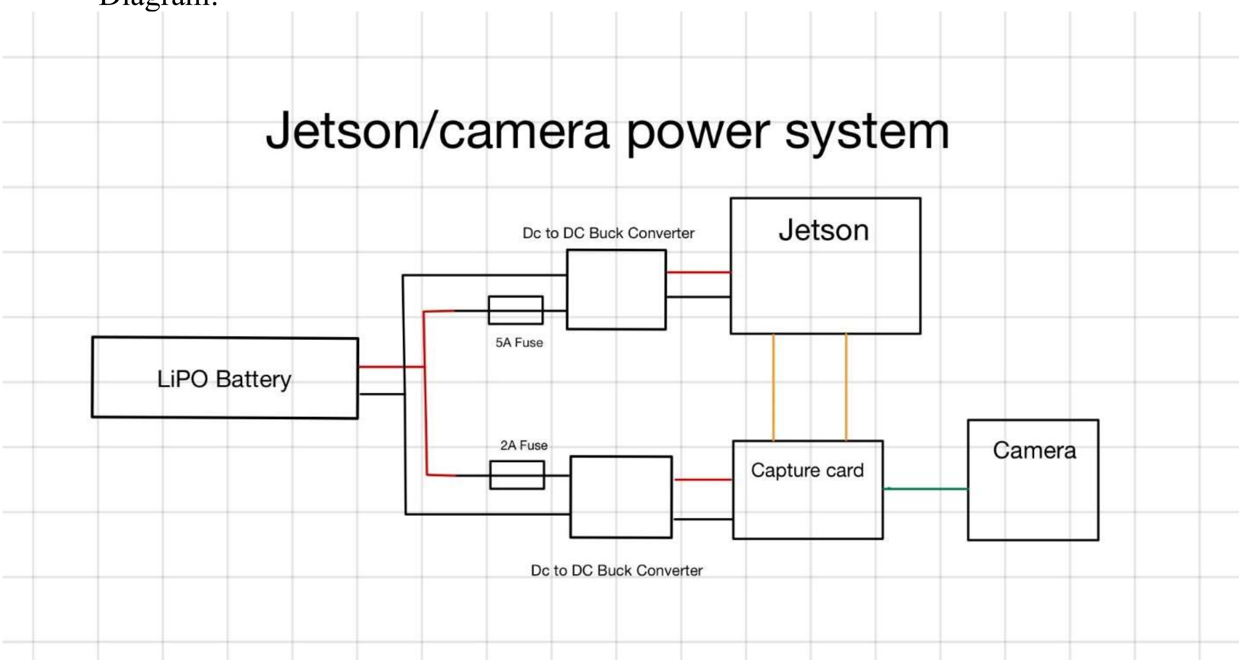

Figure 1: Jetson/Camera Power System

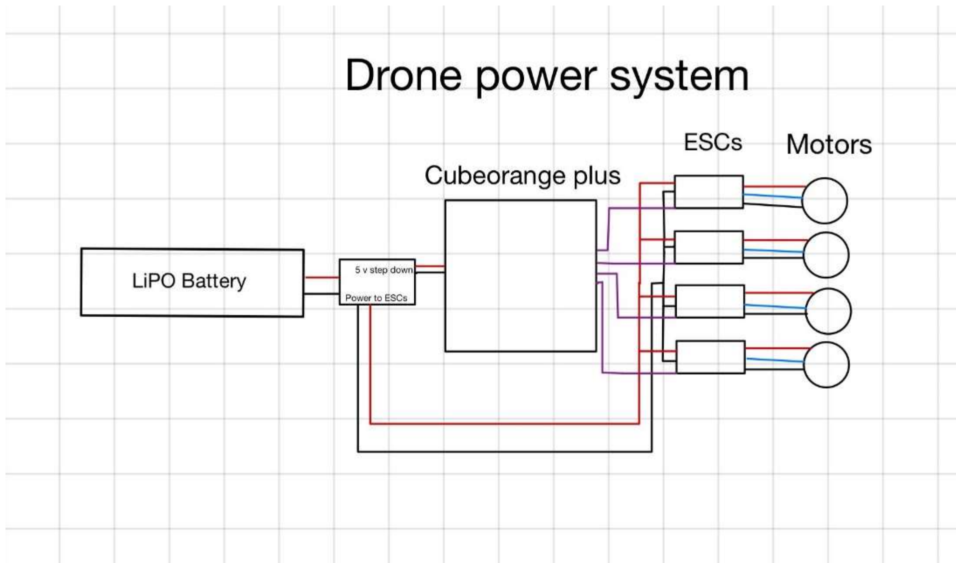

Figure 2:Drone Power System

Communication System: The communication subsystem serves as the control center for all interactions between the drone and the commands it receives. It enables both manual control through an FS-i6X RC controller and automatic control facilitated by artificial intelligence, a control system, and telemetry data.

FS-iA10B Receiver and Remote: The FS-iA10B receiver, with 10 input channels, operates at 2.4 GHz and communicates with the FlySky FS16X RC controller. The RC controller operates on a frequency from 2.408 to 2.475 GHz, requiring 4 double A batteries and featuring 10 channels with switches, axes, and dials. Integrating the Flysky FS16X controller with the Pixhawk and the receiver was essential. The controller features 10 channels and operates at 2.4 GHz. Initially, the FS-iA10B receiver struggled to read signals from the controller. It connects to the Pixhawk serially via ground, power, and signal wires. To establish proper communication, the controller's signal had to switch from PWM to PPM. The receiver then converts these Pulse Position Modulation signals to PWM format for the Pixhawk, ensuring appropriate motor response. Once communication was established, the RC controller's channels were defined by their transmission frequencies, enabling adjustments in roll, pitch, yaw, and flight modes. Three flight modes were assigned to switch 5 on the controller. The first, 'Stable Mode', is used for arming and manual takeoff, with adjustable parameters. The second, 'Altitude Hold Mode', maintains the drone's current altitude, allowing for roll, pitch, and yaw, but restricting vertical movement. The last mode, 'Land Mode', guides the drone to slowly descend and land at its current location."

Ground Control Station: The ground control station, consisting of QGroundControl and MAVPROXY, facilitates manual operation and setup. QGroundControl provides qualitative telemetry data, while MAVPROXY acts as the communication link between the quadcopter and Pixhawk.

Micro-XRCE-DDS-Agent: Data transfer between the Jetson onboard computer and the Flight Controller occurs through USB at a rate of 480 Mbps, ensuring efficient communication within the drone's system. Allows uORB messages to be sent from the Jetson to the drone and these messages can be observed.

Camera to Jetson: The ZedWrapper allows for the camera to integrate directly with ROS2. This publishes all camera topics directly into the ROS2 environment.

Telemetry Module: Module to communicate with Qgroundcontrol. Allows wireless transmission of data for in-flight processes from the ground control station

ROS2: Utilizing the Robot Operating System (ROS) offers a modular framework for robotics software development, which is essential in our project. Our current system comprises the Jetson Orin NX as the onboard computer. This computer hosts the ROS2 Humble system, which includes nodes for our custom-trained YOLO v8 object detection algorithm. Additionally, we utilize the ZedX Mini camera, integrated into ROS2 using the Zed wrapper. This integration allows us to access and utilize the camera's features fully. To control the drone, it was necessary to incorporate the Flight Controller, which runs on PX4 firmware, into the ROS2 ecosystem. With PX4 integrated within ROS2, we have developed nodes capable of sending control signals to the drone. These signals are based on inputs from the object detection algorithm, enabling the drone to follow specific objects autonomously.

Image Detection Pipeline Node Architecture: The image detection pipeline operates within the ROS2 framework using the publisher-subscriber method. The system consists of several interconnected nodes, depicted as ovals, that either publish or subscribe to data channels, known as topics. These nodes are organized into three main sections: the ZED wrapper, image processing, and object detection. The ZED wrapper, highlighted in black, publishes topics that include left and right images along with depth information from a ZED camera. For image processing, our custom nodes, shown in blue, utilize NVIDIA's accelerated software to convert images from BGR8a to BGR8 format and resize them from the original 960x600 pixels provided by the ZED wrapper to 800x512 pixels, aligning with the training size required by our model. The DNN Image Encoder node within Isaac ROS encodes image data to a format suitable for neural network inference. Efficiency is further enhanced by the TensorRT node, which optimizes tensors and neural network models to run effectively on NVIDIA GPUs. The Yolov8 decoder processes these tensors to publish outputs for object detection. Additionally, the PX4 control node manages offboard control, allowing the Jetson device to transmit control signals to motors via set points, based on the outcomes of object detection. This holistic setup ensures that all data and control signals are managed effectively within the ROS2 environment, facilitating robust image detection and response actions.

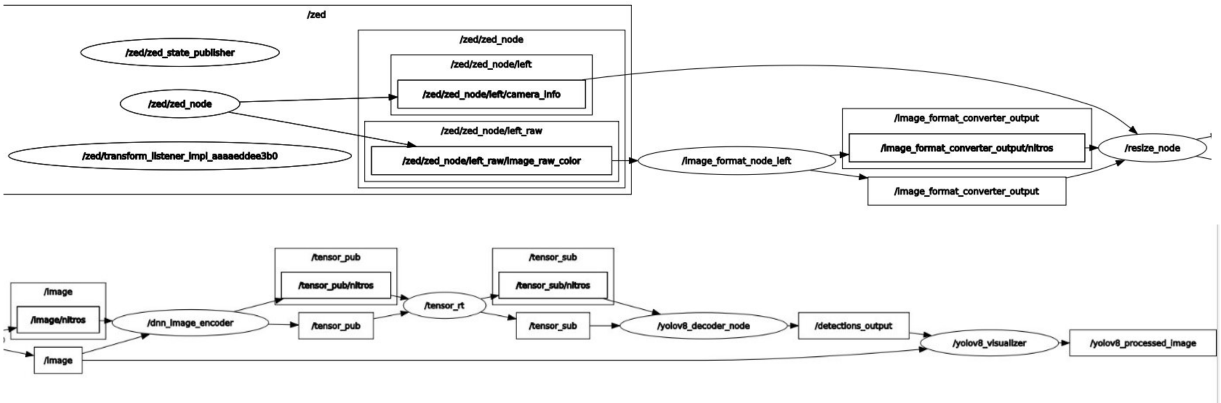

Figure 3: Full Image Processing Node System

<u>Object Detection:</u> We have chosen YOLOv8 as our object detection algorithm due to its efficient use of resources and impressive detection capabilities. Specifically, we utilize YOLOv8s, the second smallest variant in the YOLOv8 series, which contains 11.2 million parameters. This model size is optimal for our system as it balances performance with the available RAM, ensuring that we can process images rapidly without exceeding hardware limitations. YOLOv8s is designed to provide high detection accuracy while maintaining speed, making it well-suited for real-time applications where both promptness and reliability are crucial. This makes it an ideal choice for environments that require fast and accurate object detection without the need for extensive computational resources.

> <u>Training:</u> We employed transfer learning to train our object detection algorithm using a custom dataset tailored to our specific application requirements. This approach allowed us to leverage the pre-trained YOLOv8s model, which is already optimized for general object detection tasks, and adapt it to recognize objects that are unique to our operational environment. By initializing our model with weights from YOLOv8s, we significantly reduced the training time and computational resources required. Our custom dataset, comprising images from our target domain, was used to finetune the model. This method ensures that the model is not only highly accurate in detecting general objects but also exceptionally proficient in identifying and classifying the specific items relevant to our use case.

<u>Control System:</u> The control subsystem is a combination of programs loaded on the Jetson and the CubeOrange Plus, fostering data exchange to achieve tracking objectives and maintain stable flight. The control system is responsible for implementing AI-supported controllers, ensuring stability, and predicting object positions when line-of-sight vision is lost.

> <u>Equations of Motion:</u> All equations of motion and any system dynamics are used from [5]. It is important to note that we assume the camera's frame of reference is oriented in the +x direction in this system of equations. The below figure is the nonlinearized dynamics of a quadcopter system.

$$
\begin{aligned}
&\dot{\phi} = p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\
&\dot{\theta} = q[c(\phi)] - r[s(\phi)] \\
&\dot{\psi} = r\tfrac{c(\phi)}{c(\theta)} + q\tfrac{s(\phi)}{c(\theta)} \\
&\dot{p} = \tfrac{I_y - I_z}{I_x} rq + \tfrac{\tau_x + \tau_{wx}}{I_x} \\
&\dot{q} = \tfrac{I_z - I_x}{I_y} pr + \tfrac{\tau_y + \tau_{wy}}{I_y} \\
&\dot{r} = \tfrac{I_x - I_y}{I_z} pq + \tfrac{\tau_z + \tau_{wz}}{I_z} \\
&\dot{u} = rv - qw - g[s(\theta)] + \tfrac{f_{wx}}{m} \\
&\dot{v} = pw - ru + g[s(\phi)c(\theta)] + \tfrac{f_{wy}}{m} \\
&\dot{w} = qu - pv + g[c(\theta)c(\phi)] + \tfrac{f_{wz} - f_t}{m} \\
&\dot{x} = w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\
&\dot{y} = v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\
&\dot{z} = w[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)]
\end{aligned}
$$

Figure 4: Dynamic equations of the system

Architecture: The internal architecture for PX4 operates in a modular fashion. It takes in sensor data, uses that to estimate the position and attitude, and then sends that through a control algorithm structure as seen in figure 5.

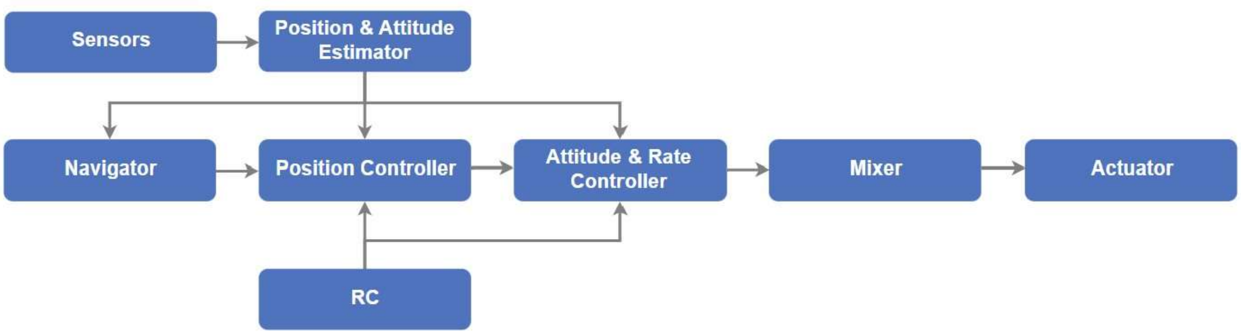

Figure 5: PX4 Control Architecture

Offboard Control: To control the system, we are using the ROS interface to publish to the "OffboardControl" topic. Through this we update the "TrajectorySetpoint" topic to publish command values for the drone to follow. Other topics allow us to publish PWM signals directly to the motors, but these were not included in the scope of this project due to time constraints.

Simulink Models: The following models are of the PX4 controller architecture. The controller takes in commands and the current state of the system. Using that it conditions the inputs through cascading PID controllers, as seen in figure 6 that feed into an attitude controller that then generates required torque signals as seen in figure 7

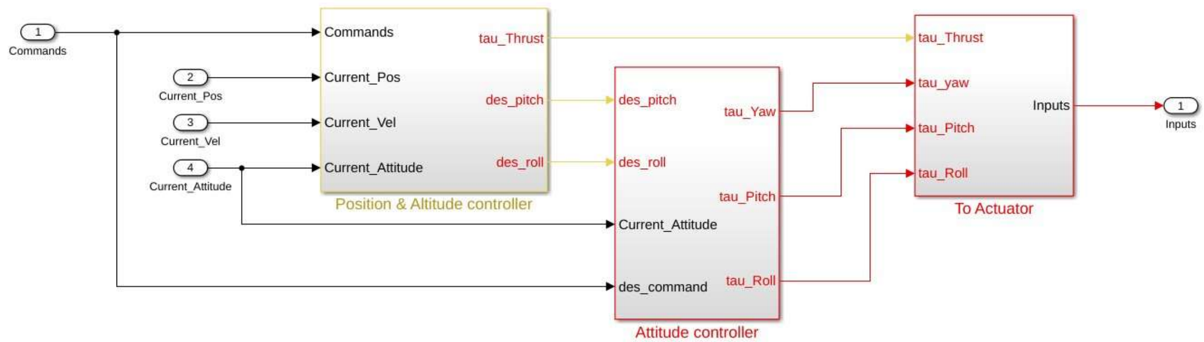

Figure 6: Simulink Control Framework

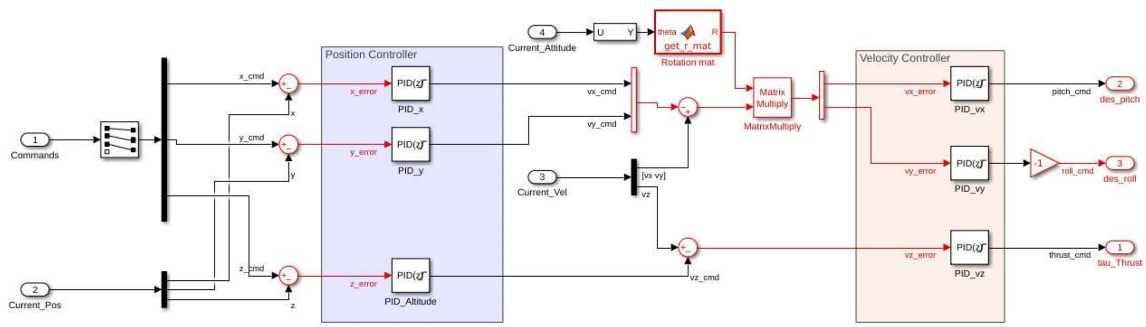

Figure 7: Attitude Control System

<u>Estimator:</u> A state estimator was designed in MATLAB with the to implement on the flight controller. Due to unreliable depth measurements, we had to use a different approach, which will be discussed in the Trajectory Estimator section. Instead, a simulation of the algorithm is used. Two simulations occur simultaneously, one for simulating the flight of the ball and then another for the camera. The error signal is defined as the difference between the line of sight of the camera and the angle the ball subtends in the camera's point of view. During the simulation, the ball's location is transformed into the camera's frame of reference, noise is added, and then the angle between the line of sight and the ball's location is then fed into an estimator which estimates the velocity and then uses that in the control signal to track the ball.

<u>Trajectory Estimator:</u> The projectile trajectory estimator begins by subscribing to the detection and depth topics provided by the image detection pipeline and the ZED wrapper. Utilizing key camera specifications such as focal length and resolution, the estimator calculates the real-world positions of detected objects. To build on this, the system incorporates system time differences (dt) to estimate the velocity of these objects. With both position and velocity determined, the estimator then applies the principles of projectile motion equations to predict future positions of the objects.

<u>System Constraints:</u> Designing a quadcopter involves navigating several constraints that collectively shape its performance, capabilities, and overall functionality.

Weight Limitations: The quadcopter's weight is limited to 3100 grams, influencing its overall flight dynamics and endurance. This will be verified by using a scale to ensure the drone is not overweight.

Power Constraints: The quadcopter's battery system is constrained to a maximum power output of 500 watts, determining the choice of motors and electronic components. This will be verified by monitoring the current draw and voltage from the flight logs during flights.

Size and Dimensions: The quadcopter's dimensions are restricted to a cubic space of 400 mm x 400 mm x 400 mm, influencing its suitability for confined spaces. This constraint will be verified by measuring the overall size of the drone.

Flight Time Requirements: The quadcopter is designed to achieve a minimum flight time of 10 minutes, determining battery capacity and energy efficiency. This will be verified by test-flying the drone until it dies and timing how long it takes until the battery is unable to provide adequate power.

Communication Range: The quadcopter has a communication range of 1.5 kilometers, influencing the choice of communication systems for remote operation. We will verify this by connecting the drone, leaving it somewhere and driving away until we get a notification from the ground station that we have disconnected and then estimate the distance using GPS on a smartphone.

V. Testing Procedures

Simulation in Hardware (SIH) is a critical methodology employed in drone development, particularly within platforms like PX4, to validate and test new changes efficiently and safely. SIH involves emulating real-world conditions in a simulated environment, allowing developers to test various scenarios without risking damage to physical hardware. In PX4, SIH integrates with hardware-in-the-loop (HITL) setups, enabling developers to run flight algorithms, test sensor integrations, and evaluate performance in a controlled environment. This approach not only accelerates development cycles but also enhances the reliability and robustness of drone systems by iteratively refining code and configurations before deployment in real-world settings. Using SIH, we can test changes we make to the offboard control scripts without having to fly the drone. We also use this method to debug when something goes wrong, and we need to test new procedures.

When testing, a certain procedure was followed to ensure optimal results. We conducted our drone testing at the Eagle Flight Research Center, situated adjacent to the ERAU Micaplex. To comply with FAA rules and regulations, our drone operations adhere to specific guidelines. FAA mandates completion of the TRUST training program and the acquisition of a certificate, which must be accessible during flight. Both Conor and Maddox have obtained TRUST certification, qualifying them as drone pilots. According to FAA regulations, drones weighing more than half a pound must be registered, unless they fall under the exception for prototype drones. When flying within an FAA-Recognized Identification Area (FRIA), such as the Eagle Flight Research Center, drone registration is not required, eliminating this obligation for our operations.

Each flight process took about 3 hours to conduct. We collected all necessary equipment including battery chargers, monitors, extension cords, and repair tools for the drone. An office was set up at the Eagle Flight Research center where the testing procedure was then started. The procedure is as follows.

1. Start the camera, image processing, and object detection algorithm. This is all done through Linux, and it prepares all the systems for flight
2. Hook the drone up to two batteries and connect it to the ground control station to prepare for flight. This is done to ensure that all elements on the hardware module can properly work together without any error.
3. The control command with the desired objective is run through the Linux computer that is connected to the drone. This offboard command is now running but it is not active.
4. The drone is armed and brought to a random position in position hold mode. Once this mode is obtained the offboard mode switch is flipped.
5. The independent code takes over the drone and brings it to a desired location. It is now actively looking for desired objects, a ball, to identify and track.
6. Once a ball enters the frame, the drone will yaw with a max yaw rate of 2.76 rad/sec to put the ball in the center of the camera frame.
7. The drone is then landed, and the process is successfully completed

VI. Results

This section will explore results obtained from the work done this semester. Most results pertain to investigations done with the manual drone operation and background setup for the drone operation. Several figures will be provided here with explanations describing what is portrayed.

To confirm that the Lidar was accurate and a reliable source of height estimation, a test was conducted by reading the output of the sensor and comparing it to a manual measurement. The results can be seen in the figure below. The largest relative error is 7% and is stable so we can consider the lidar a reliable source for height estimation.

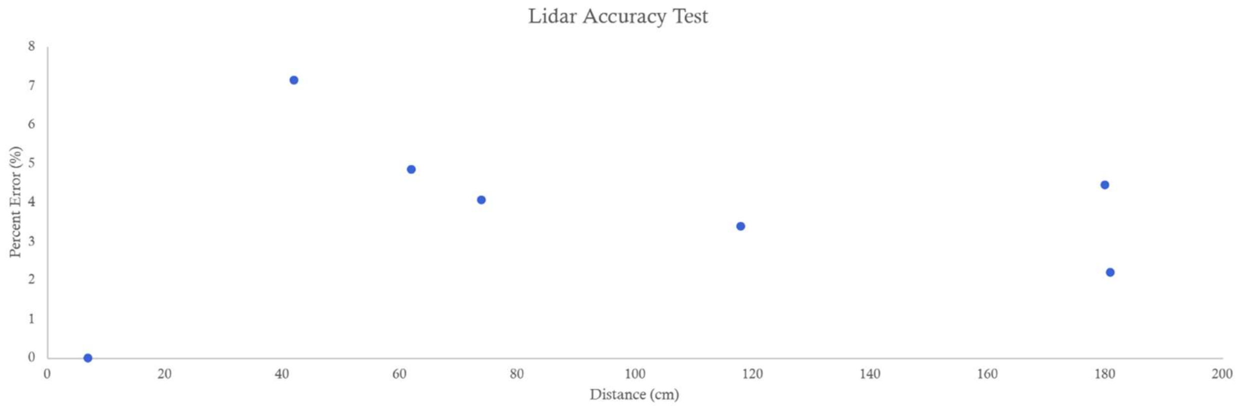

Figure 8: Lidar Test Results

The following two plots are of the height estimation. The first plot shows the GPS, lidar, and barometer estimate. As one can see, the barometer and GPS are not reliable. This is why the EKF is important because we need to be able to set the sensor fusion to emphasize the Lidar since it is the most reliable.

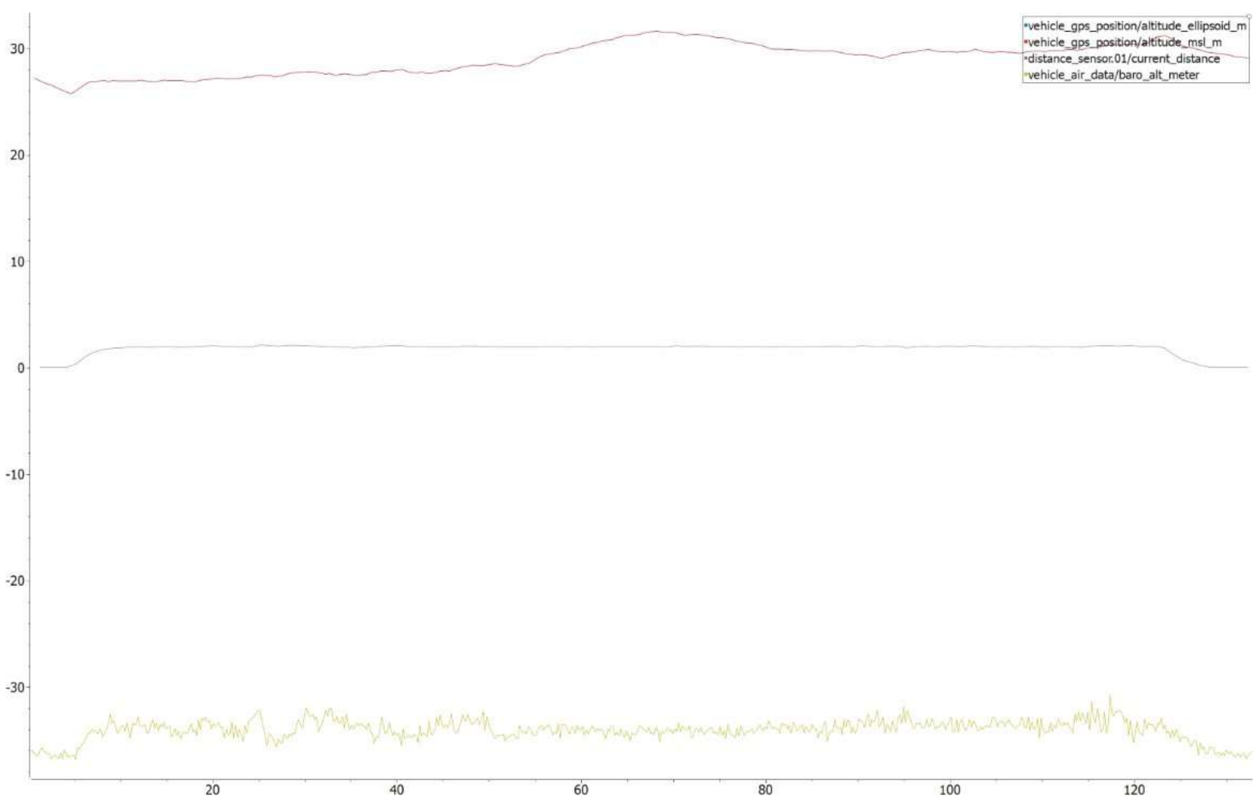

Figure 9: Lidar Height Estimate in Flight

¶

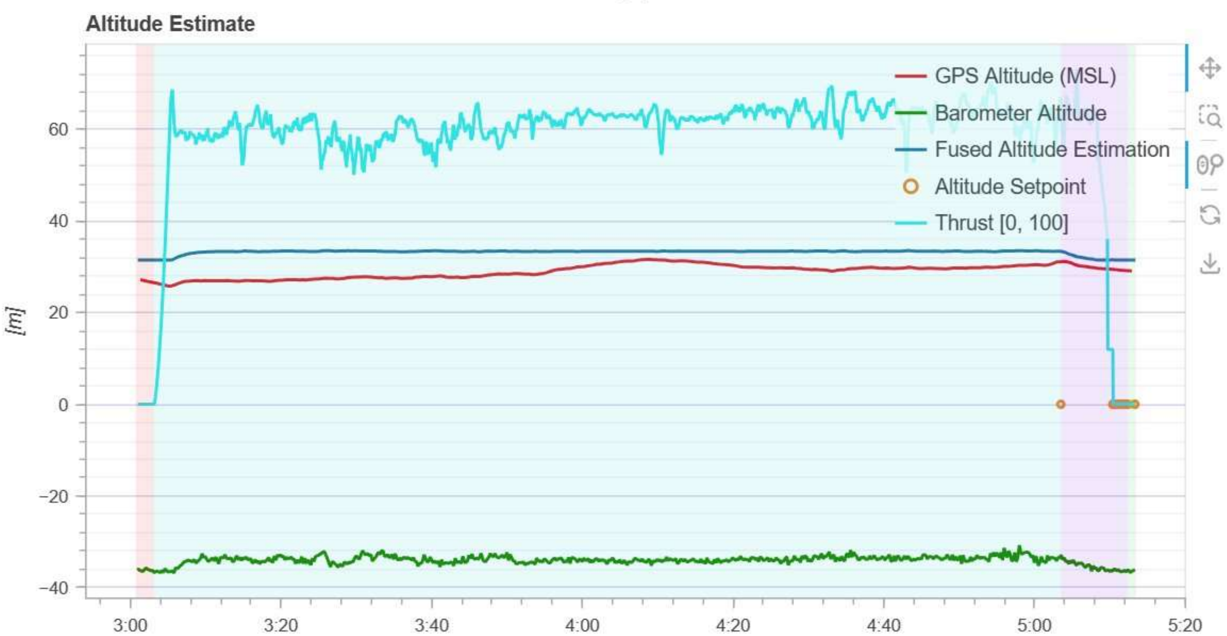

Figure 10: Other Sensor Height Estimates

The figure below was conducted to determine what the highest detection velocity of the camera would be. For this to count as a detection the drone must be facing the ball on the ground. Path in flight was measured by going frame by frame in a video to determine how long the ball was in the error to find velocity. The max velocity detected was found to be 4.13 m/s.

Distance (m)	Time (s)	Velocity (m/s)	Detected (Y/N)
5	1.21	4.13	Y
5	1.35	3.70	Y
5	0.95	5.26	N
5	1.39	3.60	Y
5	1.10	4.54	N
5	0.88	5.68	N
5	0.60	8.33	N
5	1.29	3.88	Y
5	2.56	1.95	Y
5	1.28	3.91	Y

Figure 11: Max Velocity Data

Power System:

The below figure shows the drone battery consumption metrics. The red line is the battery voltage. One can see how the battery slowly drains as the flight goes on. The grey line is the current that the drone is pulling. This is dependent on the power requirements from the control system. The magenta line is the power supplied to the flight controller, which should be a stable 5V. The light blue line is the amount of current discharged. Figure 13shows the power that is dissipated by the battery system in watts. These metrics convey a healthy power supply that meets the requirements of the power distribution system.

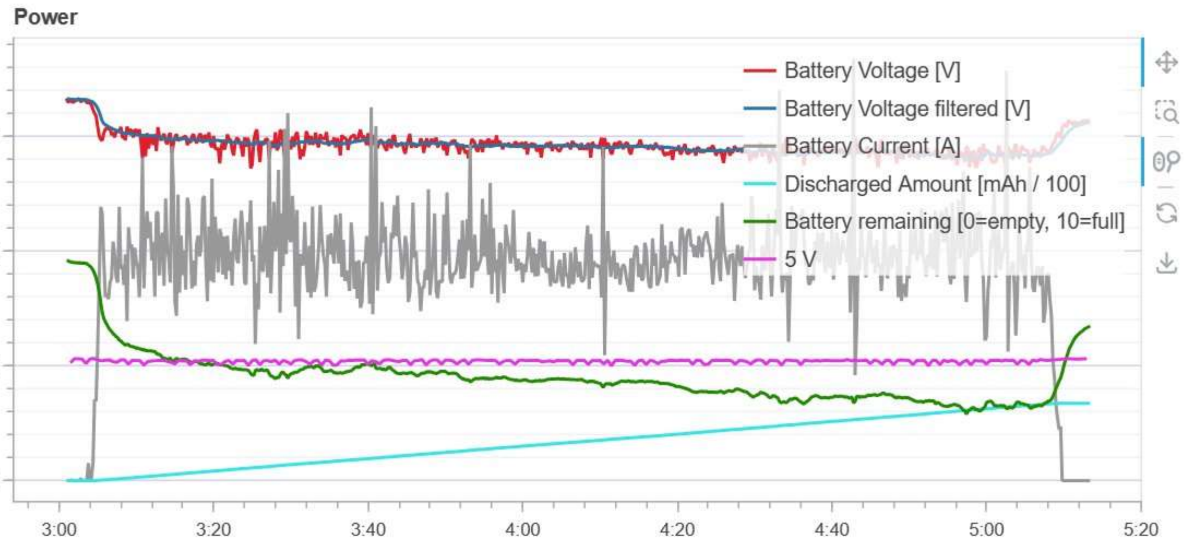

Figure 12: Drone Battery Metrics in Flight

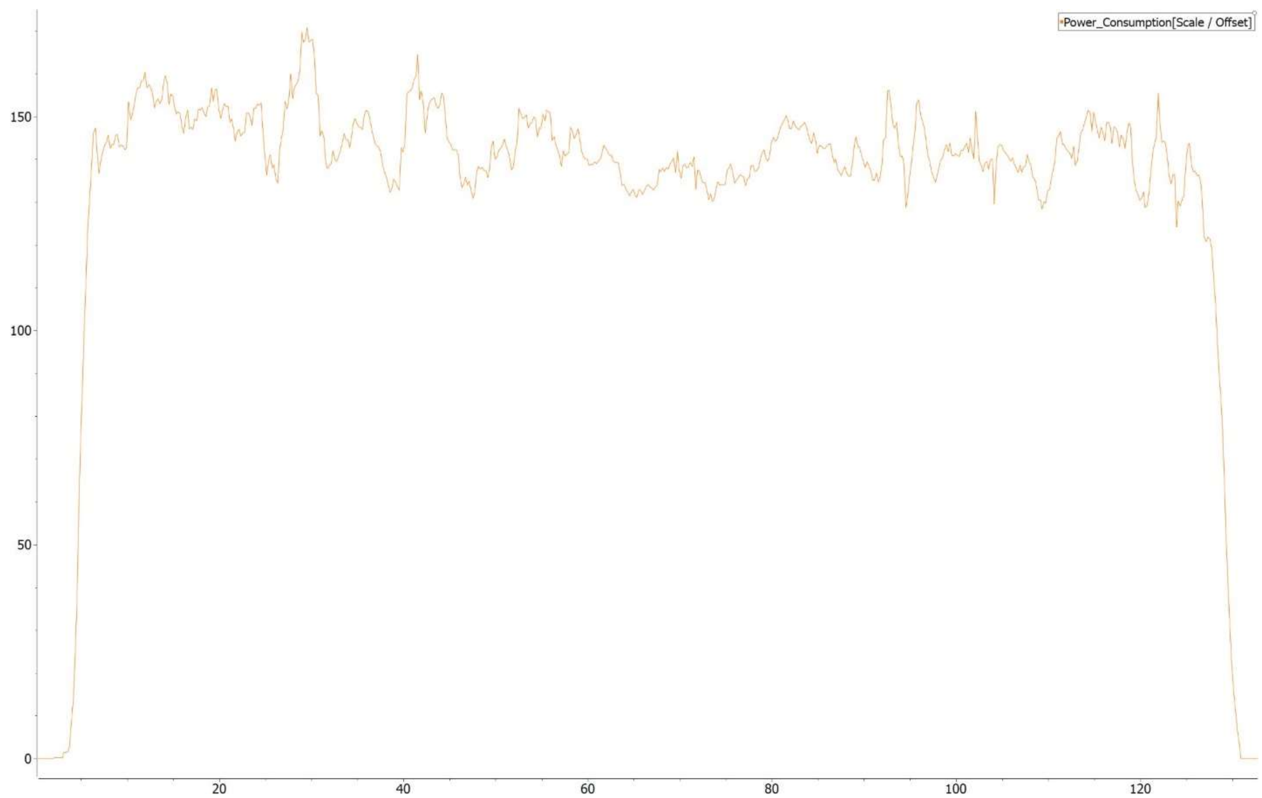

Figure 13: Power Consumption of the Drone

The figure below shows the health check of our dataset conducted using Roboflow. This check provided valuable insights into its current state and areas for improvement. Initially, our image dataset consisted of 15,566 images, including about 2,000 images from the COCO dataset, specifically from the "sports ball class," to enhance its robustness. To further adapt our dataset for YOLOv8, we resized all images to the trainable size of 800x512 pixels. We also implemented several augmentations, increasing the total number of images to 37,370. These augmentations included vertical and horizontal flips, cropping (30% zoom), adjustments in brightness (ranging from -20% to +20%) and adding noise (up to 0.3% of pixels). Despite these enhancements, the health check indicated that the 'ball' class remains underrepresented, suggesting a need for more diverse size distribution in our dataset. This insight will guide us in enriching our dataset to improve the model's robustness and accuracy across different scenarios.

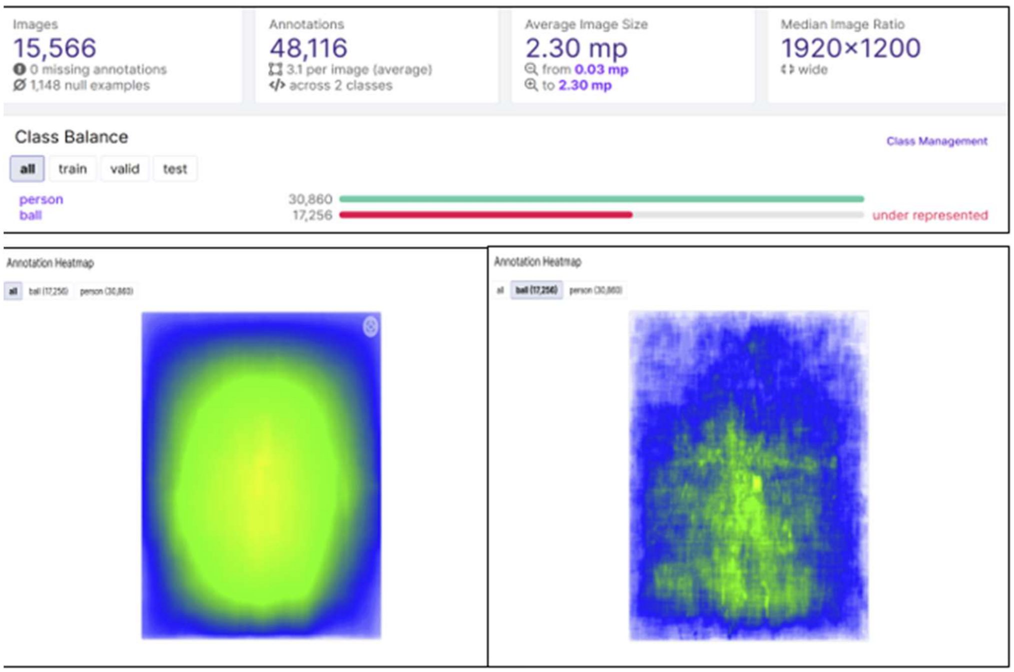

Figure 14: Health Check of Dataset

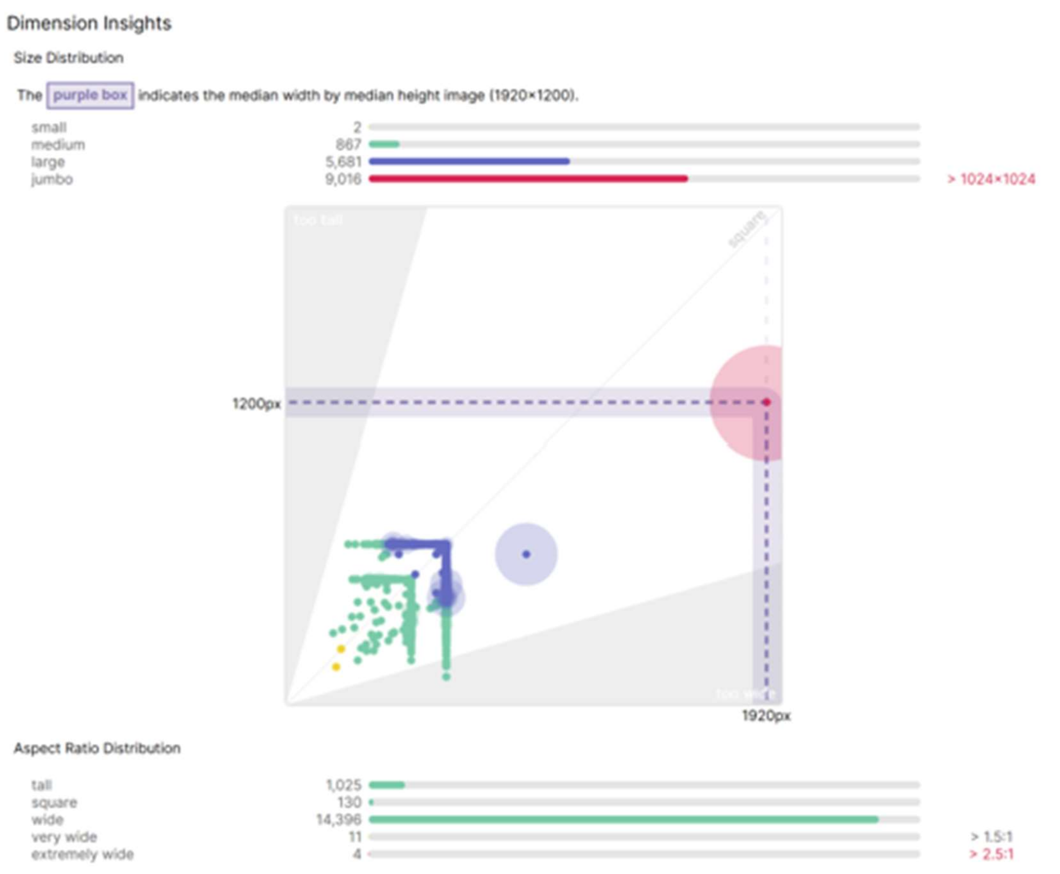

Figure 15: Dimension and Resolution Insights on Training Data

The training outcomes from our custom dataset, as revealed by the normalized confusion matrix pictured below, demonstrate distinct performance metrics for the two classes we focused on: 'ball' and 'person'. The true positive rates for these classes were 58% for 'ball' and 85% for 'person', respectively. However, the matrix also highlighted areas for improvement. Specifically, the false negative rates were notably high, with 'ball' at 41% and 'person' at 15%. Moreover, the false positive rates presented further challenges, recorded at 31% for 'ball' and a substantial 69% for 'person'.

These results suggest that while our model is more adept at identifying 'person' accurately, it struggles with precision in both classes, particularly with 'person' leading to a high rate of false alarms. To enhance the model's performance, we can consider several strategies. Firstly, increasing the diversity and representation of the 'ball' class in our training dataset could help reduce its false negatives. Additionally, refining our detection algorithm to better discriminate between 'person' and non-target objects may decrease the high false positive rate for 'person'. Implementing more targeted data augmentation techniques or adjusting the threshold settings for class predictions could also contribute to more balanced and accurate outcomes. These improvements will be crucial in optimizing our model for real-world applications where precision and reliability are key.

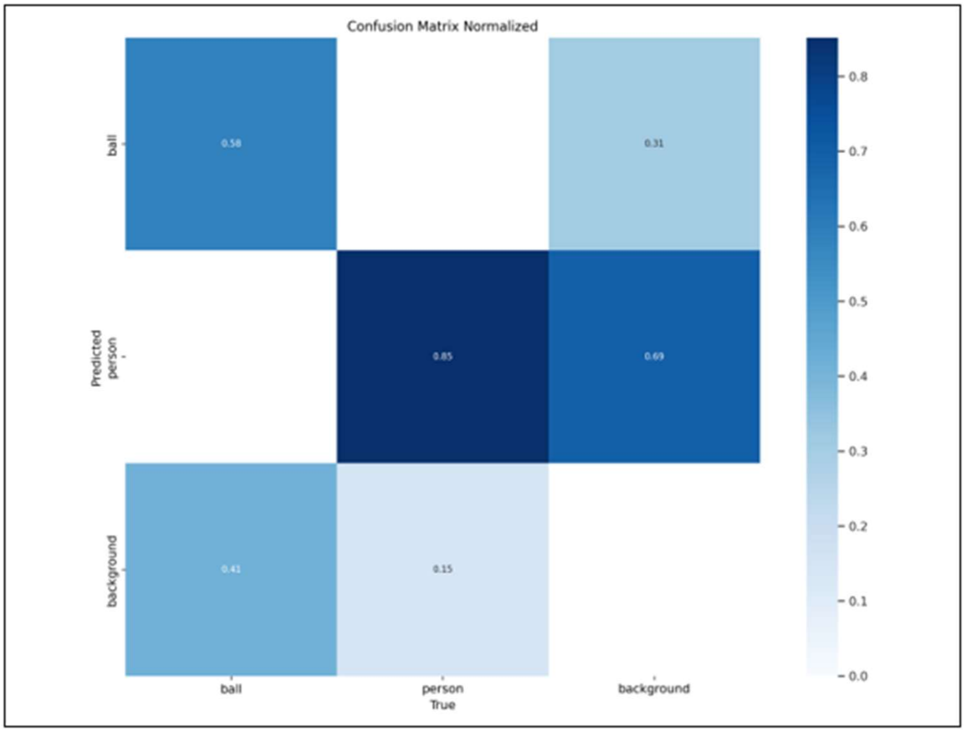

Figure 16: Confusion Matrix

The F1-confidence curve graph provides a clear depiction of the performance trade-offs for our object detection model across different confidence thresholds. The graph reveals that the optimal F1 score for both 'ball' and 'person' classes is 72% when the confidence threshold is set at 36.4%. This indicates a balanced point between precision and recall at this specific confidence level, suggesting that it effectively minimizes the rate of both false positives and false negatives. Setting the confidence threshold at 36.4% allows the model to achieve a high degree of accuracy while maintaining a reasonable detection rate, making it suitable for applications where both factors are critical. This optimal point also suggests potential areas for further tuning; by adjusting the model's sensitivity to achieve higher precision without significantly sacrificing recall, or vice versa, we can strive to push the F1 score even higher, enhancing the overall efficacy of our detection system.

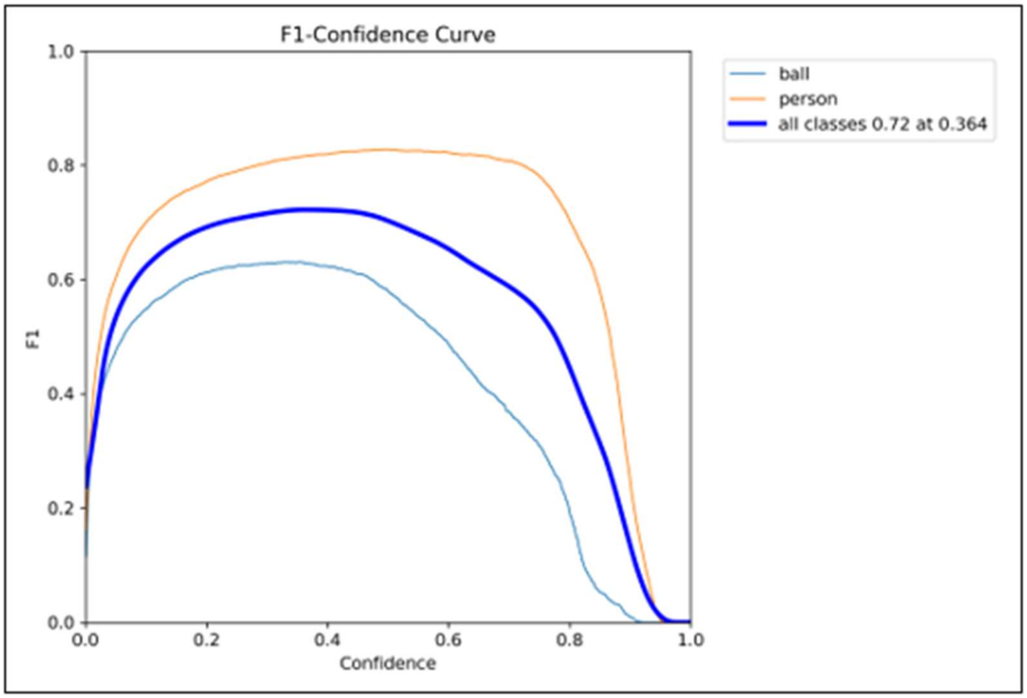

Figure 17: JF1-Confidence Curve

The figure below plots the confidence level vs the distance. The optimal distance away is 5 meters. At 5 meters we fit into the optimal confidence level of at least 60%. At 5 meters away the ball is encompassed by 4 pixels. The ball is completely lost at 9 meters where the confidence is right at 30%. There are only 3 pixels here that contain the ball. For comparison, at one meter away the ball is encompassed by 22 pixels.

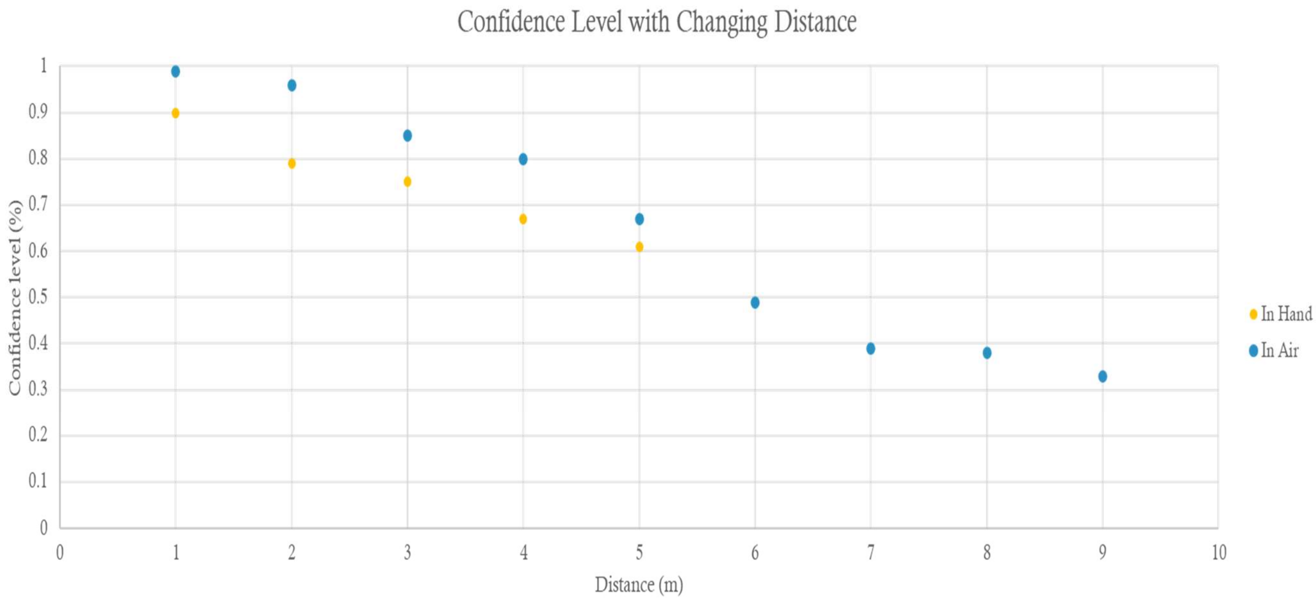

Figure 18: Confidence Level with Changing Distance

The gain table shows the applied gains for the quadcopter. These gains were very touchy and due to the cascading nature of the PID, when one changed, the system was greatly affected. PID, PI, and just P controllers were used for components of this drone.

GAIN TABLE	Proportional	Derivative	Integral
Roll Rate	1	.003	.2
Pitch Rate	1	.003	.2
Yaw Rate	1	0	.1
Roll	6.5	0	0
Pitch	6.5	0	0
Yaw	2.8	0	0
Velocity x-y	1.8	.2	.3
Velocity vertical	4	0	2
Position x-y	.95	0	0
Position vertical	1	0	0

Figure 19: Gains of the Flight Controller

The next three graphs show the yaw data throughout a flight test that was run. It can be seen how the drone continues to rotate following the path of the ball. It additionally can be seen how the step response falls within our desired tracking requirements. There is some steady state error due to it only having a proportional controller but there is an appropriate settling time and no overshoot.

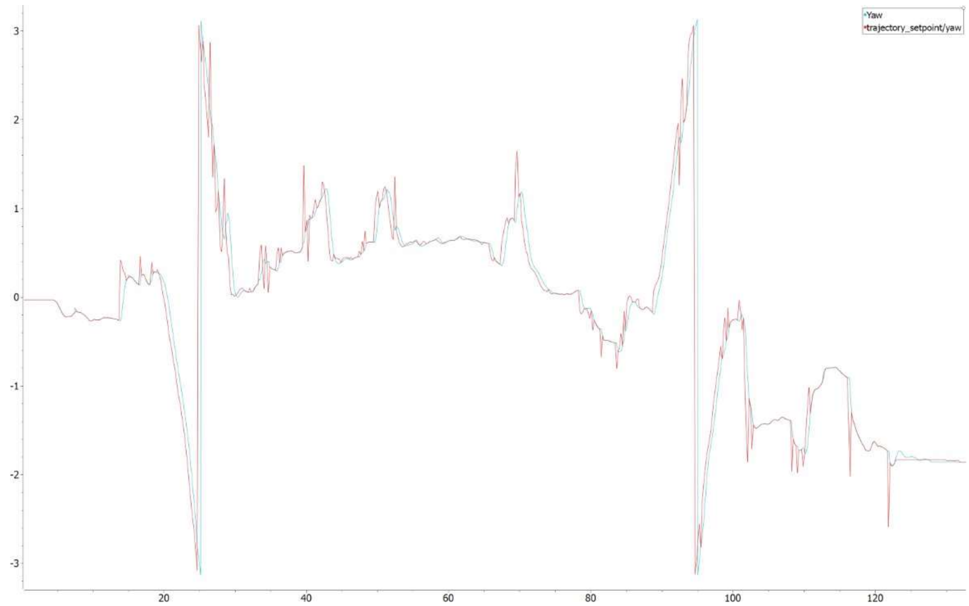

Figure 20: Yaw Setpoint

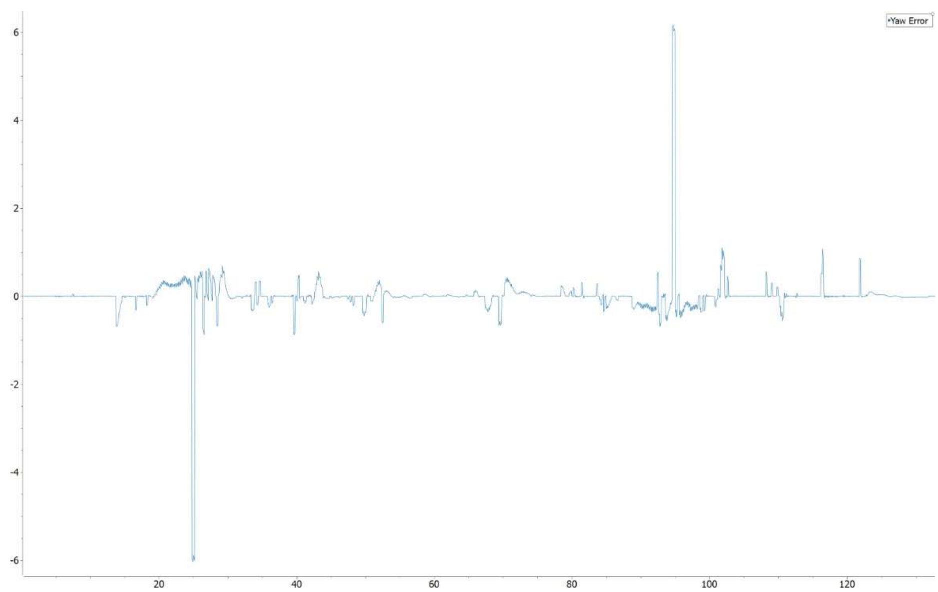

Figure 21: Yaw Angular Rate

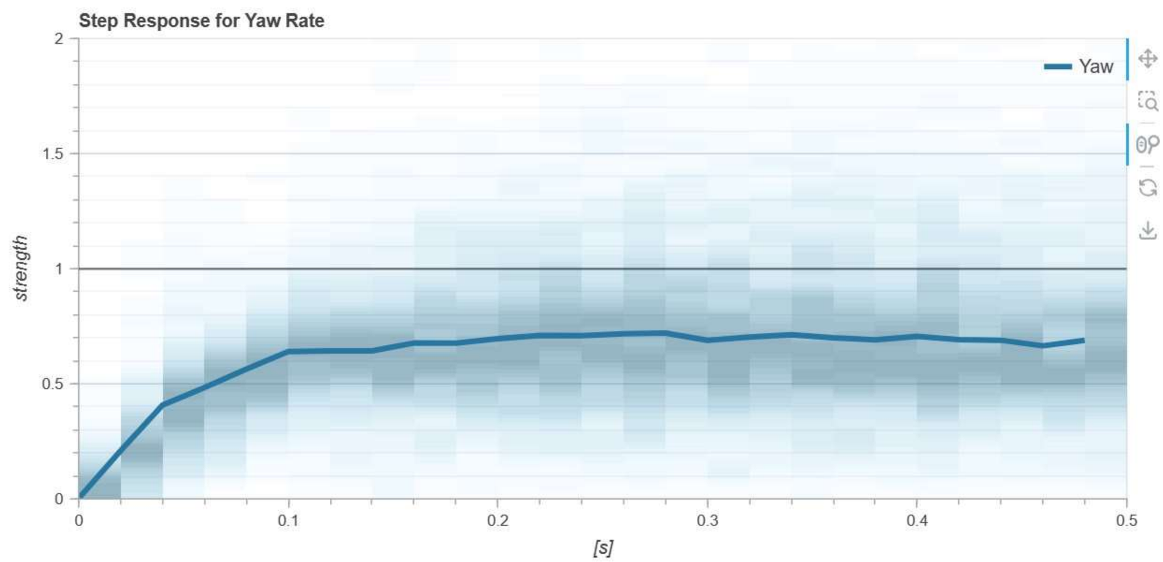

Figure 22: Yaw Step Response

The following three graphs are results of the estimator simulation. The ball starts at y= 2.5 m. The ball is imparted with 5 m/s of initial velocity in the x and z direction giving it a 45-degree launch angle. The camera is located at x=2.5 m and z = .5 m. The below figure shows the flight trajectory of the ball.

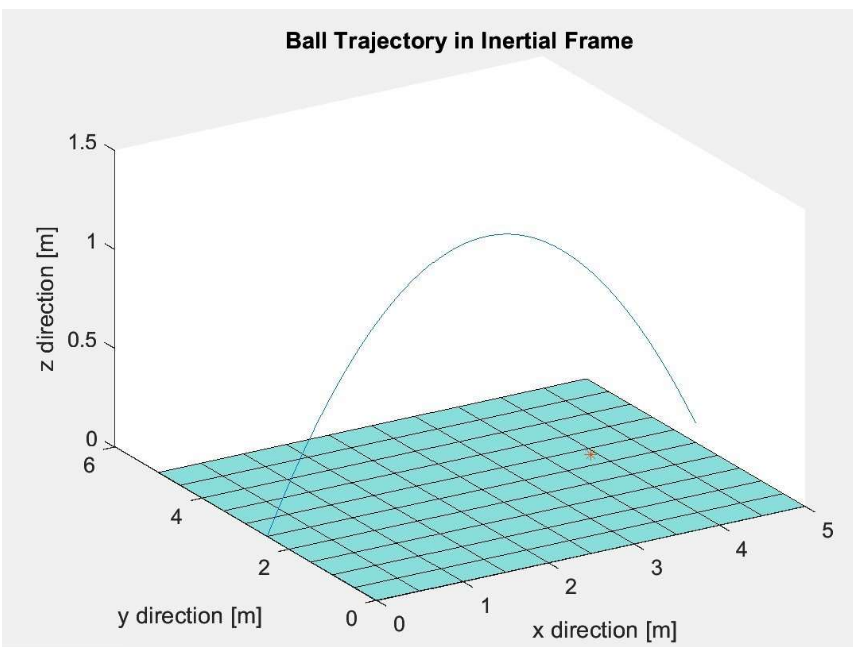

Figure 23: Ball Trajectory in MATLAB Estimator

The graph below shows the results of the control algorithm in the inertial frame. As the ball traverses the frame of the camera you can see the camera move in the CCW direction and then follow in the CW direction as the ball progresses on its trajectory.

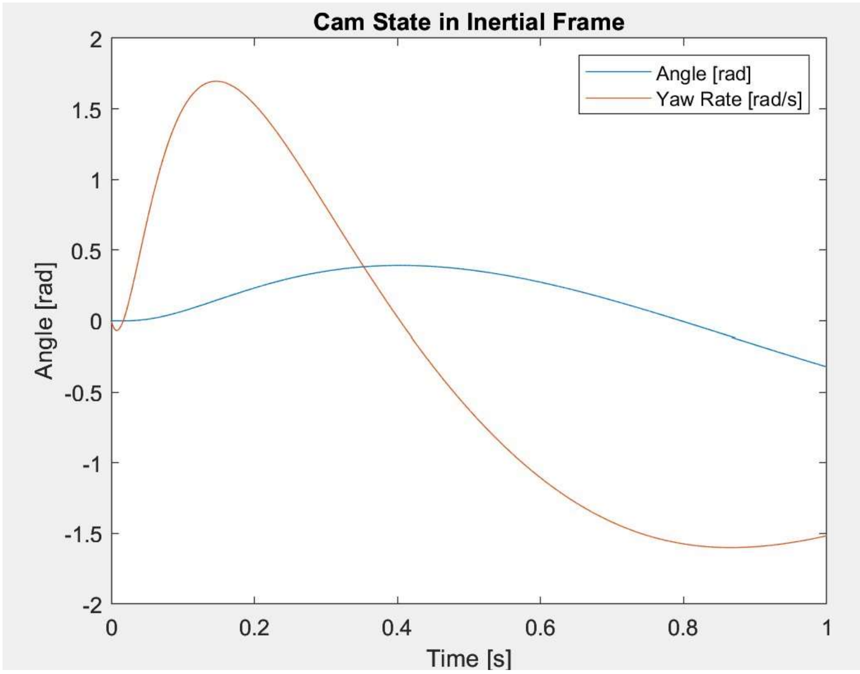

Figure 24: Camera State in Inertial frame following the ball in the simulation

The graph below shows the system in the camera-ball error space. One can see that the camera quickly converges to the trajectory of the ball and estimates the trajectory with time with only feedback from the ball's position. The angular rate follows the rate that the ball is moving across the frame.

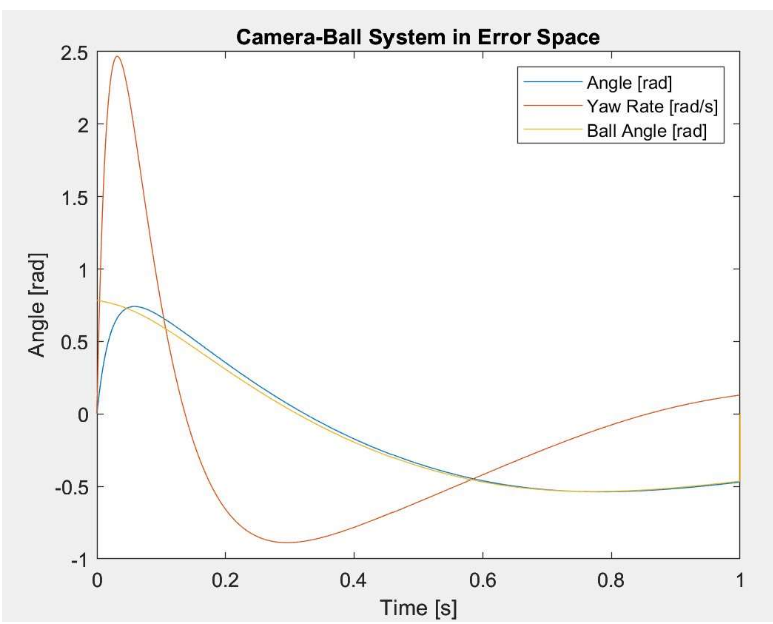

Figure 25: Camera-Ball system in error space as camera is following the ball in the simulation

The results of the projectile trajectory test, as illustrated in the diagram below, indicate that our current system experiences a 60% error rate, primarily due to inaccurate depth estimations provided by the camera. This significant level of inaccuracy stems from limitations in the camera's depth sensing capabilities, which are critical for calculating the precise real-world positions of detected objects. Initially, we had planned to enhance the depth estimation accuracy by implementing a neural depth network. However, this enhancement was not feasible due to memory constraints within our system, as the deployment of such a network requires more RAM than is currently available.

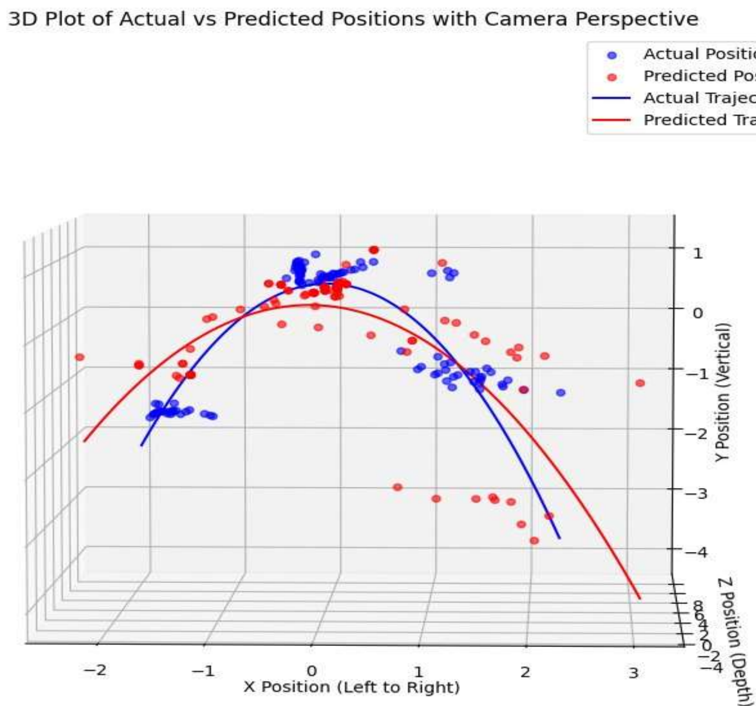

Figure 26: Trajectory Design of Live Test

Figure 27: CAD of the quadcopter body with camera attached

All our Computer-Aided Design (CAD) work has been conducted using Autodesk Inventor. To kickstart our design process, we acquired an existing CAD model of our drone body from GrabCAD.com, leveraging a pre-existing model. Ensuring precision, we cross-verified the model's dimensions with our drone body to validate its accuracy. CAD models for components such as the Jetson, ZedX Mini Camera, and Camera Capture Card were sourced from the official NVIDIA website under product information. Additionally, for components not available online, such as the battery mounting plates and the dampening plate mount, we meticulously created CAD models using Autodesk Inventor.

VII. Closing Remarks

In future work, our focus will be on implementing advanced features to enhance the capabilities and autonomy of the drone system. One key aspect will involve integrating and refining the estimator code to improve localization accuracy and stability during flight. We plan to incorporate a pixel angular size distance estimator, leveraging visual data to estimate distances to objects accurately. With this we will build a Kalman filter that will provide a stable estimate of the distance. Additionally, we aim to leverage the Robot Operating System (ROS) to directly control motors by sending PWM inputs, enabling finer control and smoother maneuvers. Another critical area of development will be obstacle avoidance within the field of view, utilizing sensor data and intelligent algorithms to navigate complex environments safely. Furthermore, we intend to implement Visual Simultaneous Localization and Mapping (VSLAM) techniques to create maps of the environment and utilize them for precise positioning information. The integration of a camera gimbal to track a target such as a ball will also be explored, allowing the algorithm to maintain the target in the center of the field of view consistently, enhancing tracking and navigation capabilities.

Throughout the development of our system, we encountered several challenges, each necessitating a specific solution to ensure efficient and accurate performance.

One of the initial issues was the integration of YOLOv8 with the ROS2 framework. To address this, we utilized the ISAAC ROS docker container, which provided a suitable environment tailored for YOLOv8, facilitating seamless integration and operation.

Another challenge was related to the image format and sizing requirements of the object detection pipeline. To adapt the images for optimal processing, we developed a custom node using NVIDIA CUDA-accelerated software. This node efficiently reformats images from BGR8A to BGR8 and resizes them from 960x600 to 800x512, aligning them with the model's specifications.

Height estimation inaccuracies were another significant hurdle, originally stemming from unreliable GPS and barometer data. We resolved this by integrating a LiDAR sensor, which offers more precise altitude readings. Consequently, we adjusted our Extended Kalman Filter (EKF) to rely predominantly on LiDAR data, greatly enhancing height estimation accuracy.
We also faced issues with the camera connection and bandwidth being too demanding for the Jetson Orin NX. To mitigate this, we reduced the bandwidth from the camera to 1.6 Gbps, which alleviated the strain on the system's processing capabilities.

Lastly, unstable depth measurements due to erratic disparity outputs from the depth camera posed a substantial challenge. Although we attempted to stabilize the depth readings using mean and standard deviation functions, these measures were only partially successful, resulting in a persistent 60% error rate in depth estimations. Plans to implement a neural depth estimator were ultimately shelved due to RAM limitations, highlighting a need for further hardware upgrades or alternative solutions to improve depth accuracy.

References

[1] "Drone Dojo Primer Course," Drone Dojo, https://dojofordrones.com/drone-programming-course/ (accessed Dec. 9, 2023).

[2] "# UORB message reference," uORB Message Reference | PX4 User Guide (main), https://docs.px4.io/main/en/msg_docs/ (accessed Dec. 9, 2023).

[3] Stereolabs, https://www.stereolabs.com/developers/release (accessed Dec. 9, 2023).

[4] "Flight review," Flight Review, https://review.px4.io/ (accessed Dec. 9, 2023)

[5] Sabatino, Franceso. "Quadrotor control: modeling, nonlinear control design, and simulation"

Project Management

I. Statement of Work

Jose Castelblanco oversees the artificial intelligence system for AIRBUD. He deals with using the Jetson and camera to ensure that neural networks are properly trained, and the communication between the companion computer and flight controller is properly established for real time control. He was instrumental in working in the Linux environment to ensure all desired outcomes occur. If there was a problem within the NVIDIA domain, he was able to quickly solve it.

Conor Metz oversees the drone operations and is team leader. He ensures that meetings and tasks get done on time and reports on weekly tasks. Additionally, he is the manual drone operator who flies the drone during test flights. He will have the RC remote during all tests and flight of the drone in either manual or automatic mode. Conor helped in all areas where additional assistance was needed to ensure project completion.

Maddox Morrison is the control leader for this project. He oversees designing a custom controller and understanding how the code on the drone operates. He analyses the sensors and how the values are interpreted by the flight controller and how to utilize the input given from the neural network subsystem. Maddox was instrumental for designing the offboard control mode and mastered C++ code in a short period of time.

II. Timeline

The following Gantt chart outlines our project's timeline, and the tasks completed this year. All major events are listed on this timeline.

TASK	ASSIGNED TO	PROGRES S	START	END
LIST OF TASK				
Define goals	All	100%	9/11/23	9/14/23
Proposal	All	100%	9/11/23	9/18/23
Budget	Jose	100%	9/12/23	10/20/23
Jetson/Liniuix Lesson	Maddox and Conor	100%	9/10/23	9/14/23

Spark Grant Proposal	Conor	100%	9/8/23	9/10/23
Pick Camera	Jose	100%	9/20/23	9/21/23
Investigate Mission Planner	Conor	100%	9/20/23	9/25/23
Connect Controller	Conor	100%	9/23/23	9/25/23
Calibrate Motors	All	100%	9/25/23	9/30/23
SITL Experimentation	Jose	100%	10/5/23	3/14/24
C Investigation	Maddox	100%	10/15/23	2/15/24
Test Fly	Conor	100%	10/15/23	4/20/24
Sensor Integration	All	100%	10/5/23	11/30/23
SRR/TRR Writing	All	100%	10/20/23	10/23/23
Optics Research	Maddox	100%	10/20/23	11/30/23
TRUST Training	Maddox and Conor	100%	10/25/23	10/30/23
New Flight Tests	Conor	100%	11/20/23	12/1/23
Weight Analysis	Conor	100%	11/25/23	11/30/23
Investigate Logs	All	100%	11/30/23	12/11/23
OrangeCube Communication	Jose	100%	12/1/23	12/7/23
Custom Control Integration	Maddox	100%	12/1/23	2/5/24
End of Semester Report	All	100%	12/1/23	12/10/23
Communication w/Jetson FC	All	100%	01/10/24	1/11/24

Create Milestones	All	100%	01/17/24	01/24/24
YOLOV8 Investigation	All	100%	01/24/24	1/25/24
Gather/Annotate Pictures	All	100%	2/09/24	2/10/24
Simulation in the Loop	All	100%	2/17/24	2/18/24
Train AI System	All	100%	2/26/2024	2/28/24
Create Custom Node	All	100%	3/1/2024	3/2/24
Complete Node Troubleshoot	All	100%	3/1/2024	3/2/24
Write Flight Scripts	All	100%	03/09/24	3/10/24
Camera to ROS Comms	All	100%	3/18/2024	3/20/24
Repair Drone	All	100%	3/25/2024	4/1/24
Lidar Implmentation	All	100%	4/08/24	4/10/24
Discovery Day	All	100%	4/08/24	4/10/24
Calculate Pixel Size	All	100%	4/08/24	4/10/24
Test Controls	All	100%	4/14/24	4/15/24
Test Full System	All	100%	4/14/24	4/15/24
Design Tests	All	100%	4/18/24	4/19/24
Conduct Tests	All	100%	4/18/24	4/19/24
Data Analysis	All	100%	4/22/24	4/23/24
Present	All	100%	4/26/24	4/26/24

Write Paper	All	100%	4/26/24	4/27/24

III. Budget

Item No.	Part	Description	price
1	Jetson Orin NX 16 GB	GPU module	$699.00
2	CubeOrange Plus	Flight Controller	$400.00
3	ZedX Mini	AI camera	$998.00
4	OVONIC 4s LiPO Battery	Battery for the flight controller and motors	$43.19
5	CNHL 6s LiPO Battery	Battery for onboard computer/camera	$78.99
6	FlySky FS-i6x	RC transmitter/receiver	$60.00
7	RFD900X-US	Telemetry Bundle	$226.00
8	Anti-gravity motors	Drone motors	$359.96
9	40 A T-Motor ESCs	ESCs for motors	$156.00
10	Here 3+ GPS Module		$175.00
11	TF-Luna Lidar	Lidar for accurate height estimation	$25.99
12	DC to DC Buck Converter	Stabilizes DC power to camera/Jetson	$7.99
		Total	$3,229.16